Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Formát SAM – reprezentace a mapování sekvencí DNA

SAM Format – Representation and Mapping of DNA Sequences

2019 Martin Kubala

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Kubala**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Formát SAM - reprezentace a mapování sekvencí DNA
SAM Format - Representation and Mapping of DNA Sequences

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je implementace programu převádějícího již namapované sekvence DNA do formátu SAM. Na vstupu programu budou sekvence DNA s již přiřazenými souřadnicemi. Ke každé sekvenci program doplní atributy definované ve specifikaci formátu SAM, především kvalitu mapování a CIGAR řetězec.

1. Popis formátu SAM a CIGAR řetězců.
2. Popis implementace a postupu pro výpočet CIGAR řetězce.
3. Experimenty a jejich vyhodnocení. Porovnání výsledných SAM souborů dosažených současnými nástroji, např. BWA, a programem sestaveným studenty.

Seznam doporučené odborné literatury:

[1] Sequence Alignment/Map Format Specification, The SAM/BAM Format Specification Working Group, online: https://samtools.github.io/hts-specs/SAMv1.pdf, 2016.
[2] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, Sequence Alignment/Map format and SAMtools. In: Bioinformatics. 25, 2009.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Vašinek**

Datum zadání: 01.09.2017
Datum odevzdání: 30.04.2019

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *26. dubna 2019*

podpis studenta

Poděkování

Rád bych poděkoval **Ing. Michal Vašinek, Ph.D.** za odbornou pomoc a konzultaci při vytváření této bakalářské práce.

Abstrakt

Cílem této bakalářské práce je implementace programu převádějícího již namapované sekvence DNA do formátu SAM. Na vstupu programu budou sekvence DNA s již přiřazenými souřadnicemi. Tyto sekvence budou zarovnány algoritmem Needleman-Wunsch a ke každé z těchto sekvencí budou doplněny povinné atributy definované ve specifikaci formátu Sam. Především CIGAR řetězec a následně kvalita mapování.

Klíčová slova: SAM, CIGAR, read, alignment, báze, kvalita

Abstract

The aim of this thesis is to implement a program that converts already mapped DNA sequences into SAM format. At the input of the program there will be DNA sequences with already assigned coordinates. These sequences will be aligned by Needleman-Wunsch algorithm and the mandatory attributes defined in the SAM format specification will be added to each of these sequences. Especially CIGAR string and consequently mapping quality.

Key Words: SAM, CIGAR, read, alignment, base, quality

Obsah

Seznam použitých zkratek

DNA	**D**eoxyribo**n**ucleic **a**cid
NCBI	**N**ational **C**enter for **B**iotechnology **I**nformation
Regex	**Reg**ular **ex**pression
SAM	**S**equence **A**lignment **M**ap
QNAME	**Q**uery template **NAME**
RNAME	**R**eference sequence **NAME**
POS	**POS**ition
MAPQ	**MAP**ping **Q**uality
RNEXT	**R**eference sequence name of the primary alignment of the **NEXT** read in the template.
PNEXT	**P**osition of the primary alignment of the **NEXT** read in the template.
TLEN	**T**emplate **LEN**gth
SEQ	**SEQ**uence
QUAL	base **QUAL**ity

Seznam ilustrací

Seznam tabulek

1 Úvod

Oblast zpracování DNA je velice široká a celá její část v dnešní době se dá popsat jako výpočetní pipeline. Ve zkratce se jedná o výpočetní elementy v sérii, jejichž výstup je vstupem následujícího výpočetního elementu. Prvním z těchto elementů je sekvenační stroj, jehož výstupem je soubor FASTQ. Tento soubor je vstupem dalšího elementu a to předzpracování, vystřihování a párování readů. Výstup tohoto segmentu je opětovně soubor FASTQ, ale už předzpracován pro další segment, a to Alignment. Touto částí se hlavně zabývá tato bakalářská práce. V prvních teoretických částech práce jsou popsány základy biologie a následně jsou popsány formáty a standardy pro vypracování této práce. Pro zarovnání budou v této práci použity dva algoritmy, a to hlavně Needleman-Wunsch [1], jenž bude hlavní součástí experimentální části a Smith-Waterman [2], který poslouží k porovnání výstupů obou algoritmů. V průběhu zarovnávaní alignmentu vůči referenčnímu genomu bude probíhat také výpočet CIGAR řetězce a následně bude proveden výpočet kvality mapování. Pro zápis výsledků do SAM souboru, jenž bude výstupem elementu, jimž se zabývá má práce, je potřeba dopočítat/doplnit zbývající povinné atributy. Následně proběhne profiltrování všech záznamů a vyřazením těch, jejichž skóre bude nižší než udaná hodnota na vstupu filtrování. Poté se všechny záznamy vypíší do SAM souboru, který je výstupem této práce. Abych doplnil celkový obraz výpočetní pipeline. Tento SAM soubor je následně vstupem dalšího elementu komprese a indexování, jehož výstupem je soubor BAM nebo BAI. Tyto soubory jsou vstupem dalšího segmentu, a to volání variant, kde dochází k profiltrování dat a jeho výstupem je soubor formátu VCF – varianty a tento soubor je vstupem pro poslední segment a to Anotace.

2 DNA

Tato část teorie je mnohdy překladem z Wikipedie, kde správnost zdrojů byla ověřena [3]. Deoxyribonucleic acid, běžně označována jako DNA (česky deoxyribonukleová kyselina, zřídka označována jako DNK), je molekula složená ze dvou řetězců, které se navzájem navinují a tím vytvářejí dvojitou šroubovici nesoucí genetické informace používané při vývoji, růstu, reprodukci a fungování všech známých živých organismů a mnoha virů s výjimkou některých nebuněčných, u kterých hraje hlavní úlohu RNA (z anglického ribonucleic acid). DNA a RNA jsou nukleové kyseliny vedle proteinů, lipidů a komplexních sacharidů (polysacharidy). Nukleové kyseliny jsou jedním ze čtyř hlavních typů makromolekul, které jsou nezbytné pro všechny známé formy života.

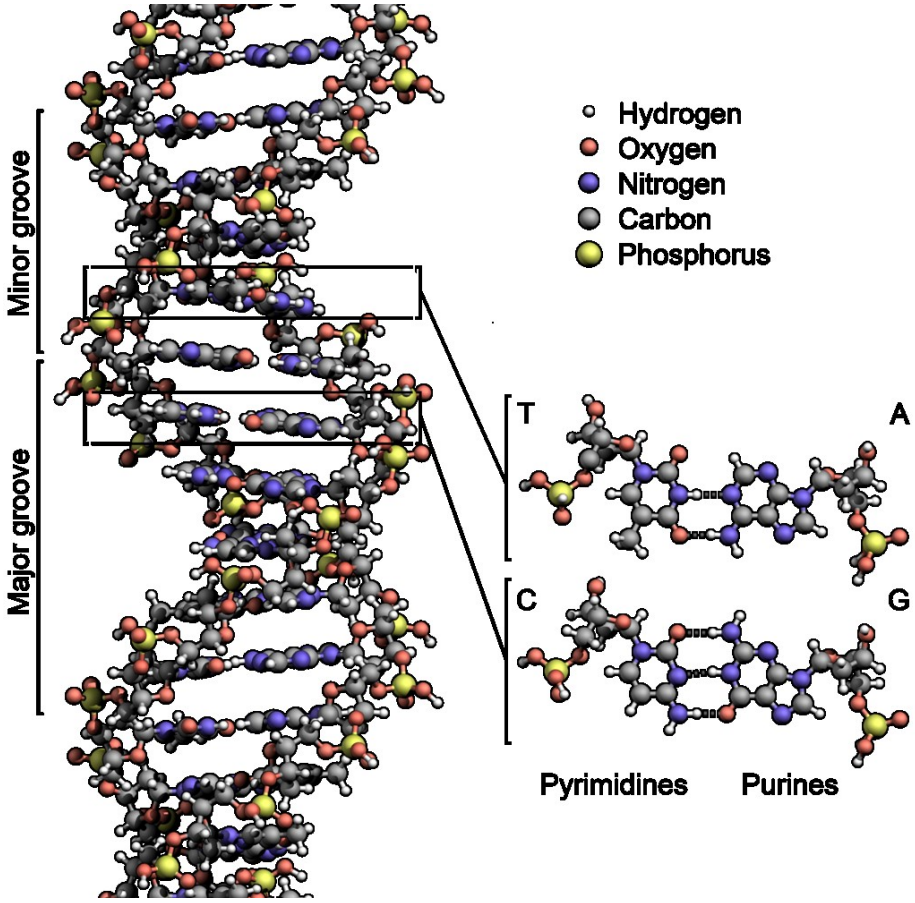
Dva DNA řetězce jsou také známé jako biologické makromolekuly neboli polymery, které se skládají z jednodušších monomerních jednotek nazývaných nukleotidy. Každý nukleotid je složen z jedné ze čtyř nukleových bází obsahujících dusík (cytosin [C], guanin [G], adenin [A] nebo thymin [T]), cukru deoxyribózy a fosfátové skupiny. Nukleotidy jsou navzájem spojeny v řetězci kovalentními vazbami mezi cukrem jednoho nukleotidu a dalším fosfátem, což vede ke střídající se páteři cukru a fosfátu. Dusíkové báze dvou oddělených polynukleotidových řetězců jsou navzájem spojeny podle pravidel párování bází (A lze spojit s T a C lze spojit s G) s vodíkovými vazbami za vzniku dvouvláknové DNA. Komplementární dusíkaté báze jsou rozděleny do dvou skupin, pyrimidiny a puriny. V DNA jsou pyrimidiny tymin a cytosin, puriny jsou adenin a guanin.

Obě vlákna dvouvláknové DNA uchovávají stejné biologické informace. Pokud dojde k oddělení obou vláken, tak se tyto biologické informace replikují. Velká část DNA (více než 98 % pro lidi) je nekódující, což znamená, že tyto sekce neslouží jako vzory proteinových sekvencí. Zároveň každé z těchto dvou vláken DNA probíhá opačným směrem a tato vlákna jsou tudíž antiparalelní. [4]

V rámci eukaryotických buněk je DNA uspořádána do dlouhých struktur nazývaných chromozomy. Před typickým dělením buněk se v procesu replikace DNA tyto chromozomy duplikují a poskytují kompletní sadu chromozomů pro každou dceřinou buňku. Eukaryotické organismy (zvířata, rostliny, houby a protisty) ukládají většinu své DNA uvnitř buněčného jádra a některé v organelách, jako jsou mitochondrie nebo chloroplasty. Naopak u prokaryot (bakterie a archea) se DNA volně uchovává pouze v cytoplazmě. [5]

V roce 1869 byla DNA poprvé izolována Friedrichem Miescherem. Molekulární struktura DNA byla poprvé identifikována v roce 1953 Francisem Crickem a Jamesem Watsonem v laboratoři Cavendish v Cambridgeské university. DNA je používána vědci jako molekulární nástroj pro zkoumání fyzikálních zákonů a teorií, jako je ergodická věta a teorie elasticity. Jedinečné materiálové vlastnosti DNA z něj činí atraktivní molekulu pro materiálové vědce a oblast inženýrů, kteří se zajímají o mikro a nano technologii. Mezi významné pokroky v této oblasti patří DNA origami a hybridní materiály založené na DNA.

Pro názornost struktura dvojité šroubovice DNA na obrázku 1, kde atomy struktury jsou barevně rozlišené elementy a podrobnější struktura dvou základních párů je uvedena vpravo dole.



Obrázek 1: Struktura DNA. [3]

2.1 Nukleotid

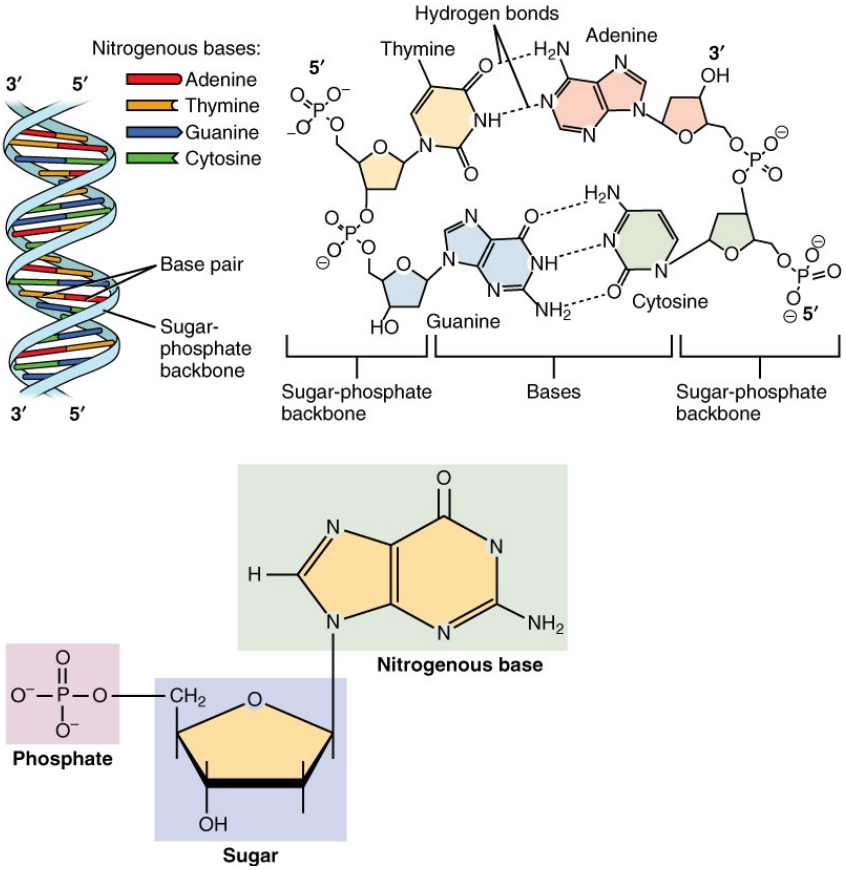
Tato část teorie (bod 2.1.x) je mnohdy překladem z Wikipedie, kde správnost zdrojů byla ověřena [6]. Nukleotid se skládá ze tří odlišných chemických podjednotek, a to z pětiuhlíkatého cukru (ribóza nebo deoxyribóza), dusíkaté báze (cytosin [C], guanin [G], adenin [A] nebo thymin [T]), které se společně nazývají nukleosid a jedna fosfátová skupina.

Nukleotidy obsahují buď purinovou nebo pyrimidinovou bázi neboli také známou jako dusíkatá bázová molekula či nukleová báze a jsou označovány jako ribonukleotidy, pokud je cukr ribózou nebo deoxyribonukleotidy, pokud je cukr deoxyribóza. Jednotlivé fosfátové molekuly opakovaně spojují molekuly cukru ve dvou sousedních nukleotidových monomerech, čímž spojují nukleotidové monomery nukleové kyseliny s dlouhým řetězcem. Tyto řetězové spojky molekul cukru a fosfátu vytvářejí řetězec páteřního řetězce pro jednu nebo dvě šroubovice. V jakémkoliv jednom vláknu, chemická orientace (směr) řetězových spojů jde od 5'-konce k 3'-konci (čteno: pátého

primárního konce k třetímu primárnímu konci), což se týká pětiuhlíkových struktur na molekulách cukru v sousedních nukleotidech. U dvojité šroubovice jsou obě vlákna orientována v opačných směrech, což umožňuje párování bází a komplementaritu mezi páry báze. To vše je nezbytné pro replikaci nebo transkripci kódované informace nalezené v DNA.

Purinové báze adenin a guanin a pyrimidinový bázový cytosin se vyskytují jak v DNA, tak v RNA, zatímco pyrimidinové báze thymin (v DNA) a uracil (v RNA) pouze v jednom. Adenin tvoří bázovou dvojici s thyminem a to dvěma vodíkovými vazbami, zatímco guanin tvoří pár s cytosinem a to třemi vodíkovými vazbami.

Pro názornost zobrazení uspořádání nukleotidů ve struktuře nukleových kyselin na obrázku 2. V levém dolním rohu je monofosfátový nukleotid, je to dusíkatá báze představující jednu stranu páru báze. Vpravo nahoře tvoří čtyři nukleotidy dvě dvojice bází, tymin a adenin (jsou spojeny dvojitou vodíkovou vazbou) a guanin a cytosin (jsou spojeny trojitou vodíkovou vazbou). Jednotlivé nukleotidové monomery jsou připojeny k řetězci na molekuly cukru a fosfátu, čímž tvoří dvě páteře (dvojitá šroubovice) nukleové kyseliny, které jsou uvedeny v levém horním rohu. [7] [8] [9]

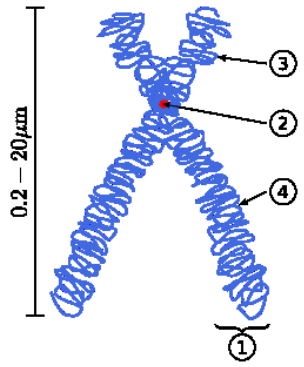


Obrázek 2: Struktura Nukleotidu. [6]

2.2 Chromozom

Chromozomem je DNA molekula s částí nebo celým genetickým materiálem (genomem) organismu. Obvykle jsou chromozomy viditelné pod světelným mikroskopem, a to pouze tehdy, kdy buňka prochází metafází buněčného dělení. Než se ale tak stane, tak se každý chromozóm jednou zkopíruje a kopie se připojí k originálu centromerem, což vede ke struktuře ve tvaru písmene X (obrázek 3 pod odstavcem), pokud je centromer umístěn ve středu chromozomu nebo do dvojité struktury, pokud je centromer umístěn v blízkosti jednoho z konců. Původní chromozom a jeho kopie jsou nyní nazývány sesterskými chromatidy. Během metafáze je struktura tvaru X nazývána metafázovým chromozómem. [10] [11]

Na obrázku 3 je zobrazeno schéma replikovaného a kondenzovaného metafázového eukaryotického chromozomu. (1) Chromatid – jedna ze dvou identických částí chromozomu po S-fázi. (2) Centromer – místo, kde se dotýkají dva chromatidy. (3) Krátké raménko (p). (4) Dlouhé raménko (q).



Obrázek 3: Eukaryotický chromozom. [11]

2.2.1 Eukaryotický chromozom

Chromozomy v eukaryotách se skládají z chromatinových vláken, která jsou vyrobena z nukleozomů (histonových oktamerů s částí vlákna DNA připojeného a obaleného kolem něj). Chromatinová vlákna jsou obalena bílkovinami do kondenzované struktury nazývané chromatin. Chromatin je přítomen ve většině buněk s několika výjimkami, např. červené krvinky a umožňuje velmi dlouhým molekulám DNA, aby se vešly do buněčného jádra. Během buněčného dělení chromatin dále kondenzuje za vzniku mikroskopicky viditelných chromozomů. Struktura chromozomů se mění v buněčném cyklu. Během buněčného dělení se chromozomy replikuji, rozdělují a úspěšně přecházejí do jejich dceřiných buněk tak, aby byla zajištěna jejich genetická rozmanitost a přežití jejich potomstva. Chromozomy můžou existovat buď to jako duplikované nebo neduplikované. Neduplikované chromozomy jsou jednoduché dvojité šroubovice, zatím co duplikované chromozomy obsahují dvě identické kopie (chromatidy nebo sesterské chromatidy) spojené centromerem.

Eukaryoty (buňky s jádry, jako jsou ty, které se nacházejí v rostlinách, houbách a zvířatech) mají několik velkých lineárních chromozomů obsažených v jádře buňky. Každý chromozom má jeden centromer s jednou nebo dvěma rameny vyčnívajícími z centromeru, i když za většiny okolností nejsou tyto ramena vidět jako takové. Navíc většina eukaryot má malý kruhový mitochondriální genom a některé eukaryoty mohou mít další malé kruhové nebo lineární cytoplazmatické chromozomy. V jaderných chromozomech eukaryot existuje nekondenzovaná DNA v polo-uspořádané struktuře, kde je zabalena kolem histonů (strukturní proteiny), tvořící kompozitní materiál nazvaný chromatin.

2.2.2 Lidský chromozom

Chromozomy u lidí lze rozdělit do dvou typů, a to jako autozomy (chromosom/y těla) a alosomy (pohlavní chromozómy). Některé genetické rysy jsou spojeny s pohlavím člověka a jsou předány prostřednictvím pohlavních chromozomů. Autozomy obsahují drtivou většinu genetických dědičných informací. Všechny chromozomy se při dělení chovají stejně. Lidské buňky mají 23 párů chromozomů (22 párů autozomů a jeden pár pohlavních chromozomů), což dává celkově 46 chromozomů na buňku. Kromě toho mají lidské buňky mnoho stovek kopií mitochondriálního genomu. Sekvenování lidského genomu poskytlo mnoho informací o každém z chromozomů. Tabulka 1 sestavuje statistiky chromozomů na základě informace lidského genomu Sanger Institutu ve Vertebrate Genome Annotation (VEGA) databázi. [12] Počet genů je odhad, protože je částečně založen na genetických předpovědích. Celková délka chromozomu je odhadem založeným na odhadu velikosti neosekvenované oblasti.

Tabulka 1: Lidské chromozomy. [12]

Chromozom	Geny	Celkem bázových párů	% bází	Sekvenované bázové páry	% sekvenovaných bázových párů
1	2000	247,199,719	8.0	224,999,719	91.02 %
2	1300	242,751,149	7.9	237,712,649	97.92 %
3	1000	199,446,827	6.5	194,704,827	97.62 %
4	1000	191,263,063	6.2	187,297,063	97.93 %
5	900	180,837,866	5.9	177,702,766	98.27 %
6	1000	170,896,993	5.5	167,273,993	97.88 %
7	900	158,821,424	5.2	154,952,424	97.56 %
8	700	146,274,826	4.7	142,612,826	97.50 %
9	800	140,442,298	4.6	120,312,298	85.67 %
10	700	135,374,737	4.4	131,624,737	97.23 %
11	1300	134,452,384	4.4	131,130,853	97.53 %
12	1100	132,289,534	4.3	130,303,534	98.50 %
13	300	114,127,980	3.7	95,559,980	83.73 %
14	800	106,360,585	3.5	88,290,585	83.01 %
15	600	100,338,915	3.3	81,341,915	81.07 %
16	800	88,822,254	2.9	78,884,754	88.81 %
17	1200	78,654,742	2.6	77,800,220	98.91 %
18	200	76,117,153	2.5	74,656,155	98.08 %
19	1500	63,806,651	2.1	55,785,651	87.43 %
20	500	62,435,965	2.0	59,505,254	95.31 %
21	200	46,944,323	1.5	34,171,998	72.79 %
22	500	49,528,953	1.6	34,893,953	70.45 %
X (pohlavní chromozom)	800	154,913,754	5.0	151,058,754	97.51 %
Y (pohlavní chromozom)	50	57,741,652	1.9	25,121,652	43.51 %
Celkem	**21,000**	**3,079,843,747**	**100.0**	**2,857,698,560**	**92.79 %**

2.3 FASTA formát

Tato část teorie (bod 2.3.x) je mnohdy překladem z Wikipedie, kde správnost zdrojů byla ověřena [13]. FASTA formát je textovým formátem, který reprezentuje nukleotidové nebo aminokyselinové (proteinové) sekvence. V těchto sekvencích jsou nukleotidy nebo aminokyseliny reprezentování za použití jednoznakového kódu. Ve formátu je také umožněno, aby názvy a komentáře předcházeli sekvence. Tento formát vychází ze softwarového balíčku FASTA, ale nyní se stal téměř universálním standardem v oblasti bioinformatiky a biochemie. Obvykle se také do FASTA formátu ukládá referenční genom. Díky jednoduchosti formátu FASTA je usnadněna manipulace a analýza sekvencí pomocí nástrojů pro zpracování textu a skriptovacích jazyků jako je programovací jazyk R, Python, Ruby a Perl. [14]

2.3.1 Záhlaví

Řádek záhlaví (známý také jako řádek popisu, hlavičky či identifikátoru), který začíná znakem '>' udává název a/nebo jedinečný identifikátor sekvence a může také obsahovat další informace. V původním formátu Pearson FASTA se může po záhlaví vyskytnout jeden nebo více komentářů, které se oddělí pomocí středníku na začátku řádku. Některé databáze a aplikace bioinformatiky nerozpoznávají tyto komentáře a nesledují specifikaci NCBI FASTA. Příklad FASTA souboru s více sekvencemi na obrázku 4.

```
>SEQUENCE_1
MTEITAAMVKELRESTGAGMMDCKNALSETNGDFDKAVQLLREKGLGKAAKKADRLAAEG
LVSVKVSDDFTIAAMRPSYLSYEDLDMTFVENEYKALVAELEKENEERRRLKDPNKPEHK
IPQFASRKQLSDAILKEAEEKIKEELKAQGKPEKIWDNIIPGKMNSFIADNSQLDSKLTL
MGQFYVMDDKKTVEQVIAEKEKEFGGKIKIVEFICFEVGEGLEKKTEDFAAEVAAQL
>SEQUENCE_2
SATVSEINSETDFVAKNDQFIALTKDTTAHIQSNSLQSVEELHSSTINGVKFEEYLKSQI
ATIGENLVVRRFATLKAGANGVVNGYIHTNGRVGVVIAAACDSAEVASKSRDLLRQICMH
```

Obrázek 4: Příklad sekvencí FASTA formátu. [13]

2.3.2 Reprezentace sekvence

Sekvence jako taková je zobrazena pod řádkem záhlaví. Sekvence mohou být proteinové sekvence nebo sekvence nukleových kyselin a mohou obsahovat mezery nebo znaky zarovnání. Sekvence jsou reprezentovány ve standardu např. IUPAC kódu [15] aminokyselin (tabulka 2) a nukleových kyselin (tabulka 3). Pro tuto práci jsou hlavně důležité nukleové kyseliny z tabulky 3.

Číslice nejsou povoleny, ale v některých databázích se používají k označení pozice v pořadí.

Tabulka 2: Kódy aminokyselin. [15]

1-znak	3-znaky	popis
A	Ala	Alanine
R	Arg	Arginine
N	Asn	Asparagine
D	Asp	Aspartic acid
C	Cys	Cysteine
Q	Gln	Glutamine
E	Glu	Glutamic acid
G	Gly	Glycine
H	His	Histidine
I	Ile	Isoleucine
L	Leu	Leucine
K	Lys	Lysine
M	Met	Methionine
F	Phe	Phenylalanine
P	Pro	Proline
S	Ser	Serine
T	Thr	Threonine
W	Trp	Tryptophan
Y	Tyr	Tyrosine
V	Val	Valine
B	Asx	Aspartic acid or Asparagine
Z	Glx	Glutamine or Glutamic acid
X	Xaa	Any amino acid

Tabulka 3: Kódy nukleových kyselin. [15]

kód	popis
A	Adenine
C	Cytosine
G	Guanine
T	Thymine
U	Uracil
R	Purine (A or G)
Y	Pyrimidine (C, T, or U)
M	C or A
K	T, U, or G
W	T, U, or A
S	C or G
B	C, T, U, or G (not A)
D	A, T, U, or G (not C)
H	A, T, U, or C (not G)
V	A, C, or G (not T, not U)
N	Any base (A, C, G, T, or U)

2.3.3 Přípona souboru

FASTA formát nemá žádný standard přípony souboru pro textový soubor obsahující sekvence. Následující tabulka 4 zobrazuje každé rozšíření a jeho příslušný význam.

Tabulka 4: Přípony souboru FASTA. [13]

přípona	**význam**	**poznámka**
fasta	obecná FASTA	Jakýkoliv soubor FASTA.
fna	FASTA nukleové kyseliny	Používá se obecně k určení nukleových kyselin.
ffn	FASTA nukleotid genových oblastí	Obsahuje kódovací oblasti genomu.
faa	FASTA aminokyseliny	Obsahuje aminokyselinové sekvence.
frn	FASTA nekodované RNA	Obsahuje nekódované oblasti RNA pro genom v DNA abecedě, např. tRNA, rRNA

2.4 FASTQ formát

Formát FASTQ je textovým formátem, kde se ukládá sekvence a Phred quality do jednoho souboru do kterého může kdokoliv nahlédnout a přečíst veškerá data. To je také důvodem, proč se k Phred skóre přičítá 33, tak aby odpovídalo prvnímu čitelnému znaku v ASCII. Tento formát je stručný a kompaktní. FASTQ formát byl poprvé ve větší míře použit v Sanger Institute, a proto díky tomu nyní obvykle používáme Sanger specifikaci a formát FASTQ. Příklad dat v souboru FASTQ na obrázku 5. [16]

```
@EAS54_6_R1_2_1_413_324
CCCTTCTTGTCTTCAGCGTTTCTCC
+
;;3;;;;;;;;;;;;7;;;;;;;88
@EAS54_6_R1_2_1_540_792
TTGGCAGGCCAAGGCCGATGGATCA
+
;;;;;;;;;;7;;;;;-;;;3;83
@EAS54_6_R1_2_1_443_348
GTTGCTTCTGGCGTGGGTGGGGGGG
+EAS54_6_R1_2_1_443_348
;;;;;;;;;;9;7;;.7;393333
```

Obrázek 5: Příklad dat v FASTQ souboru. [16]

2.4.1 Specifikace formátu

Níže jsou pravidla pro popsání syntaxe:

- <fastq>, <blok> atd. představují neterminální symboly.
- Červeně značeny jsou operátory podobné regex.
- '\ n' znamená nový řádek.

Syntaxe samotná:

<fastq> := <blok>+

<blok> := @<seqname>\n<seq>\n+[<seqname>]\n<qual>\n

<seqname> := [A-Za-z0-9_.:-]+

<seq> := [A-Za-z\n\.~]+

<qual> := [!-~\n]+

Podmínky:

- Sekvence <seqname>, která následuje po '+', je volitelná, ale pokud se objeví hned za '+', měla by být totožná s <seqname> po '@'.
- Délka <seq> odpovídá délce <qual>. Každý znak v <qual> představuje phread kvalitu odpovídajícího nukleotidu v <seq>.
- Pokud je kvalita phred **Q**, což je ne-záporné celé číslo, odpovídající znak kvality **q** lze vypočítat pomocí následujícího vzorce v C++:

  q = (char) ((**Q** <= 93 ? **Q** : 93) + 33);

  , kde (char) převede celé číslo na znak založený na ASCII tabulce.
- Opačným procesem pro získání kvality phred **Q** je vložení znaku kvality **q** do následujícího vzorce v C++:

  Q = (int) (**q** – 33);

2.5 Phread quality score

Skóre kvality Phred je měřením kvality identifikace nukleotidů generovaných automatickým sekvenováním DNA. Tradičně je kvalita Phred definována na základě čtení báze. Každé čtení báze je odhadem pravého nukleotidu. Je to náhodná proměnná, která může nabývat hodnoty nesprávného nukleotidu. Pravděpodobnost nesprávného čtení báze se nazývá pravděpodobnost chyby. Pokud pravděpodobnosti chyby báze přiřadíme znak **P**, tak Phred kvalitu báze **Q** vypočteme pomoci vzorce:

$$\boldsymbol{Q} = -10\, log_{10}\, \boldsymbol{P}$$

Opačný výpočet pro získání pravděpodobnosti chyby báze **P** z Phred kvality báze **Q** provedeme pomoci vzorce:

$$\boldsymbol{P} = 10^{\frac{-\boldsymbol{Q}}{10}}$$

Hodnoty skóre Phred jsou logaritmicky spojeny s pravděpodobnostmi chyb. V tabulce 5 jsou uvedeny příkladné hodnoty.

Tabulka 5: Phred skóre logaritmicky spojeno s pravděpodobnostmi chyb. [18]

Skóre kvality Phred	Pravděpodobnost špatného přečtení báze	Přesnost čtení báze
10	1 z 10	90 %
20	1 z 100	99 %
30	1 z 1000	99.9 %
40	1 z 10,000	99.99 %
50	1 z 100,000	99.999 %
60	1 z 1,000,000	99.9999 %

Příklad použití:

Pokud máme pravděpodobnost chyby čtení báze 1 z 1,000, tak do vzorce za **P** dosadíme číslo 0.001.

$$\boldsymbol{Q} = -10\, log_{10}\, 0.001$$

Výsledkem je **Q** = 30. Tudíž máme potvrzeno, že kvalita Phred odpovídá pravděpodobnosti špatného přečtení báze. Pro opačný postup dosadíme kvalitu Phred **Q** = 30 a získáme tím pravděpodobnost chyby báze **P**.

$$\boldsymbol{P} = 10^{\frac{-30}{10}}$$

Výsledkem je **P** = 0.001. Což odpovídá tabulkové hodnotě 1 z 1000. [17] [18]

2.6 Mapping quality score

Mapping quality skóre je v podstatě pravděpodobnost, že je read zarovnán na nesprávném místě neboli poloha mapování tohoto readu je nesprávná. Pro výpočet mapping quality skóre je použit vzorec:

$$\boldsymbol{Qs} = 4 + (3 - \boldsymbol{k}')(\overline{\boldsymbol{q}} - 14) - 4.343 \log \boldsymbol{p1}(3 - \boldsymbol{k}', 28)$$

Kde $\boldsymbol{k}'$ je počet všech neshod alignmentu a $\overline{\boldsymbol{q}}$ je průměr kvality. Pro genom člověka je **p1** (0, 28) ≈ 0,2, **p1** (1, 28) ≈ 0,05 a **p1** (2, 28) ≈ 0,03. Vzhledem k tomu, že kvalita je zaznamenána v logu,

různé přiblížení zde nebude mít velký vliv na přesnost kvality mapování. Proto v této práci je použita pouze první část vzorce a to:

$$\boldsymbol{Qs} = 4 + (3 - \boldsymbol{k}')(\overline{\boldsymbol{q}} - 14)$$

Když $\boldsymbol{k}'$ = 0 neshod alignmentu a $\overline{\boldsymbol{q}}$ = 38, tak výsledkem je hodnota 76. Pokud je výsledná hodnota **Qs** > 60, tak **Qs** = 60. Pokud je výsledná hodnota **Qs** < 0, tak **Qs** = 0. [19]

2.7 CIGAR řetězec

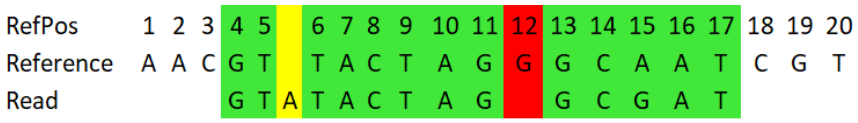
Read sekvence, která je zarovnána na referenční sekvenci, může mít další báze, které nejsou v referenční sekvenci nebo read sekvenci můžou chybět báze, které jsou v referenční sekvenci. Řetězec CIGAR je sekvence délek báze a související operace. Používají se k označení takových věcí, jako které báze jsou zarovnány (buď shoda / neshoda) s referenční sekvencí nebo jsou z referenční sekvence vymazány nebo vložené báze, které v referenční sekvenci nejsou. [20]

Na obrázku 6 je uvedený příklad referenční sekvence a Readu, který bude k referenční sekvenci zarovnán. Atribut RefPos znamená pozici referenční sekvence, Reference znamená bází referenční sekvence a Read je načtená sekvence po sekvenování.

RefPos	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Reference	A	A	C	G	T	T	A	C	T	A	G	G	G	C	A	A	T	C	G	T
Read	G	T	A	T	A	C	T	A	G	G	C	A	T							

Obrázek 6: Příklad referenční sekvence a readu.

Na obrázku 7 je příklad zarovnání readu na referenční sekvenci z obrázku 6.



Obrázek 7: Příklad zarovnané read sekvence na referenční sekvenci.

Z obrázku 7 získáme data POS: 4 a CIGAR: 2M1I6M1D5M, kde POS označuje pozici, kde začíná zarovnání readu na referenční sekvenci. CIGAR nám říká, že první dvě báze readu jsou zarovnány se referenční sekvencí (2M). Další báze v readu neexistuje v referenční sekvenci (1I). Poté šest bází readu je zarovnáno s referenční sekvencí (6M). Další báze referenční sekvence neexistuje v read sekvenci (1D) a nakonec je dalších pět read bází zarovnáno s referenční sekvencí (5M). Poznámka k poloze 15, kde je báze readu jiná než referenční báze se stále počítá jako M, protože se zarovnává na této poloze.

2.7.1 Extended CIGAR

Pro formát SAM je použita rozšířená verze CIGAR, kde je použito více operátorů, které jsou popsány v tabulce 6.

Tabulka 6: CIGAR operátory. [21]

Operátor	Popis
M	shoda nebo neshoda báze
I	Vložení do referenční sekvence
D	Smazání z referenční sekvence
N	Vynechání regiónu z referenční sekvence
S	soft clipping (oříznuté sekvence v read sekvenci)
H	hard clipping (oříznutí sekvence neexistujících sekvencí v read sekvenci)
P	padding (přidání prázdných znaků pro hezčí vizualizaci zarovnání)
=	shoda báze
X	neshoda báze

- Suma délek operátorů M/I/S/=/X se rovná délce read sekvence.
- H operátor může být přítomný pouze jako první a / nebo poslední operace.
- S může mít mezi sebou pouze H operace a konec řetězce CIGAR.

2.8 SAM formát

SAM znamená Sequence Alignment Map formát. Jedná se o textový formát oddělovaný tabulátorem, který se skládá ze sekce záhlaví, která je nepovinná a ze sekce alignmentů, která je povinná. Je-li záhlaví k dispozici, musí být zapsána před alignmenty. Řádky záhlaví začínají vždy znakem @, zatímco alignmenty ne. Každý řádek alignmentu má 11 povinných atributů základních informací jako je např. pozice mapování a pak se skládá z volitelného počtu nepovinných atributů obsahujících specifické informace zarovnání. [21]

2.8.1 Terminologie SAM

Předloha (z anglického template) – Sekvence DNA / RNA, jejíž část je sekvenována na sekvenčním stroji nebo sestavena ze surových sekvencí.

Segment – Sousední sekvence nebo subsekvence.

Read – Surová sekvence, která pochází ze sekvenování. Read se může skládat z několika segmentů. Pro sekvenční data jsou ready indexovány podle pořadí, ve kterém jsou sekvennovány.

Lineární alignment (z anglického linear alignment) – Zarovnání readu na jednu referenční sekvenci, která může obsahovat vkládání (z anglického Insert), mazání (z anglického Delete), přeskakování (z anglického Skips) a oříznutí (z anglického Clipping), ale nemusí zahrnovat změny směru (tj. jedna část alignmentu na dopředním řetězci a další část alignmentu na reverzním řetězci). Lineární alignment může být reprezentován v jediném SAM záznamu.

Chimérický alignment (z anglického chimeric alignment) – Alignment readu, který nelze reprezentovat jako lineární alignment. Chimérický alignment je reprezentován jako soubor lineárních alignmentů, které nemají velké překrytí. Typicky jeden z lineárních alignmentů v chimérickém alignmentu považován za „reprezentativní“ alignment a ostatní alignmenty jsou označeny jako „doplňkové“ a rozlišují se doplňkovým flagem alignmentu. Všechny záznamy SAM v chimérickém alignmentu mají stejný QNAME a stejné hodnoty pro flagy 0x40 a 0x80 (popsáno v sekci 2.8.3). Rozhodnutí, který lineární alignment je reprezentativní je libovolné.

Read alignment – Lineární alignment nebo chimérický alignment, který je úplnou reprezentací alignmentu readu.

Vícenásobné mapování (z anglického Multiple mapping) – Správné umístění readu může být nejednoznačné, např. kvůli opakování. V tomto případě může být více read alignmentů pro stejný read. Jeden z těchto alignmentů je považován za primární. Všechny ostatní alignmenty mají nastavený sekundární alignment flag v SAM záznamu, který je reprezentuje. Všechny záznamy SAM mají stejný QNAME a stejné hodnoty pro flagy 0x40 a 0x80. Obvykle primárním alignmentem je určen alignment s nejlepším zarovnáním, ale rozhodnutí může být libovolné. [21]

2.8.2 Sekce záhlaví

Každý řádek záhlaví začíná znakem @ následovaným dvoupísmenovým kódem. V záhlaví je každý řádek oddělován tabulátorem a na rozdíl od řádku @CO, každé datové pole následuje formát TAG:HODNOTA kde tag je dvoupísmenkové označení definující formát a obsah hodnoty. Tagy označeny znakem '*' jsou vyžadovány jako např. každý @SQ řádek záhlaví musí mít SN a LN pole. Tagy, které lze použít budou popsány níže. [21]

První z Tagů je @HD, řádek záhlaví, a pokud je přítomen, tak je na prvním řádku. Jeho další datové tagy jsou popsány v tabulce 7.

Tabulka 7: SAM – Záhlaví, tagy @HD. [21]

Tag	Popis
VN*	Verze formátu. Akceptovaný formát: /^[0-9]+\.[0-9]+$/.
SO	Pořadí alignmentů. Platné hodnoty: neznáme (výchozí), netříděné, název příkazu a koordináty. Pro třídění koordinátů je hlavním třídicím klíčem pole RNAME v pořadí definovaným pořadím @SQ tagů v záhlaví. Vedlejším třídicím klíčem je pole POS. Pro alignmenty kde jsou si RNAME a POS rovny, je pořadí náhodné. Všechny alignmenty s * v RNAME poli následují alignmenty s jinou hodnotou, ale jinak jsou v libovolném pořadí.
GO	Seskupování alignmentů, indikuje, že podobné alignment záznamy jsou seskupeny dohromady, ale soubor nemusí být nutně seřazen. Platné hodnoty: žádné (výchozí), dotaz (alignmenty jsou seskupeny podle QNAME), a odkaz (alignmenty jsou seskupeny podle RNAME / POS).

Dalším tagem je @SQ, slovník referenčních sekvencí. Pořadí @SQ určuje pořadí seřazených alignmentů. Jeho další datové tagy jsou popsány v tabulce 8.

Tabulka 8: SAM – Záhlaví, tagy @SQ. [21]

Tag	Popis
SN*	Název referenční sekvence. Tagy SN a individuální názvy AN musí být ve všech řádcích SQ@ odlišné. Hodnota tohoto pole je použita v záznamech alignmentu v polích RNAME a RNEXT. Regulární výraz: [!-)+-<>-~][!-~]*
LN*	Délka referenční sekvence. Rozsah: $[1, 2^{31} - 1]$
AH	Indikuje, že tato sekvence je alternativní místo. Hodnota je místo v primární sestavě, pro kterou je tato sekvence alternativou ve formátu „chr:začátek-konec“, „chr“ (když je znám) nebo „*“ (pokud není znám), kde „chr“ je sekvence v primární sestavě. Nesmí být přítomen na sekvencích v primární sestavě.
AN	Alternativní názvy referenčních sekvencí. Seznam alternativních jmen oddělených čárkou, kterou mohou být použity nástroji, při odkazování na tuto referenční sekvenci. Tyto alternativní názvy nejsou použity kdekoliv jinde v SAM souboru, zejména nesmí být použity v záznamech alignmentu v polích RNAME nebo RNEXT. Regulární výraz: jméno(,jméno)* kde jméno je [0-9A-Za-z][0-9A-Za-z*+.@ |-]*
AS	Identifikátor sestavení genomu.
M5	MD5 kontrolní součet sekvence.
SP	Druh.
UR	URI sekvence. Tato hodnota začíná jedním ze standartních protokolů jako je např. http nebo ftp. Pokud nezačíná s jedním z těchto protokolů, předpokládá se, že jde o systémovou cestu např. c:\<cesta>.

Dalším tagem je @RG. Skupina Readu. Neuspořádání vícero @RG řádků je povoleno. Jeho další datové tagy jsou popsány v tabulce 9.

Tabulka 9: SAM – Záhlaví, tagy @RG. [21]

Tag	Popis
ID*	Identifikátor skupiny readu. Každý řádek @RG musí mít unikátní ID. ID hodnota je použita v RG tagu alignment záznamů. Tato hodnota musí být unikátní mezi všemi skupinami readů v sekci záhlaví. ID read skupin můžou být upraveny při slučování souborů SAM za účelem zpracování konfliktů.
CN	Název sekvenčního centra produkující read.
DS	Popis.
DT	Datum vytvoření dat (ISO8601 datum nebo datum/čas).
FO	Pořadí toku. Pole nukleotidových bází, které odpovídají nukleotidům použitým pro každý tok každého readu. Více-bázové toky jsou zakódovány ve formátu IUPAC a nenukleotidové toky různými jinými znaky. Formát: /*|[ACMGRSVTWYHKDBN]+/
KS	Pole nukleotidových, které odpovídají klíčové sekvenci každého readu.
LB	Knihovna.
PG	Programy použity pro zpracování skupiny readu.
PI	Předpokládaný medián vkládané velikosti.
PL	Platforma/technologie použité pro výrobu readu. Platné hodnoty: CAPILLARY, LS454, ILLUMINA, SOLID, HELICOS, IONTORRENT, ONT a PACBIO.
PM	Model platformy. Volný text poskytující další podrobnosti o použité platformě/technologii.
PU	Jednotka platformy (např. flowcell-barcode.lane pro Illumina nebo slide pro SOLid). Unikátní identifikátor.
SM	Vzorek. Použije název fondu, kde je fond sekvencován.

Jeden z posledních tagů je @PG. Program. Jeho další datové tagy jsou popsány v tabulce 10.

Tabulka 10: SAM – Záhlaví, tagy @PG. [21]

Tag	Popis
ID*	Identifikátor záznamu programu. Každý řádek @PG musí mít unikátní ID. Hodnota ID je použita v alignment tagu PG a tagů PP ostatních řádků @PG. PG ID můžou být upraveny při slučování souborů SAM za účelem zpracování konfliktů.
PN	Název programu.
CL	Příkazový řádek.
PP	Předchozí @PG ID. Musí se shodovat s jiným @PG ID tagem záhlaví. @PG záznamy mohou být řetězeny pomocí PP tagu, přičemž poslední záznam v řetězci nemá žádný PP tag. Tento řetězec definuje pořadí programů, které byly aplikovány na alignment. Hodnoty PP můžou být upraveny při slučování souborů SAM za účelem zpracování konfliktů PG ID. První PG záznam v řetězci (tj. Ten, který je označen PG tagem v SAM záznamu) popisuje poslední program, který pracoval na SAM záznamu. Další PG záznam v řetězci popisuje další program z posledních, který pracoval na SAM záznamu. Pro PG ID na SAM záznamu není vyžadováno, aby odkazovalo na nejnovější PG záznam v řetězci. Může odkazovat na jakýkoliv PG záznam v řetězci, což znamená, že SAM záznam byl zpracován programem v daném PG záznamu a program (y) odkazované přes tag PP.
DS	Popis.
VN	Verze programu.

Posledním tagem je @CO. Jednořádkový textový komentář. Je povoleno mít více @CO neuspořádaných řádků. Tento tag nemá další datové tagy.

2.8.3 Sekce alignmentu – povinná pole

Ve SAM formátu, každý řádek alignmentu typicky představuje lineární alignment segmentu. Každý řádek má 11 povinných datových polí. Tato pole se vždy zobrazují ve stejném pořadí a musí být přítomna, ale jejich hodnoty mohou být 0 nebo * (v závislosti na poli), pokud nejsou příslušné informace k dispozici. V následující tabulce 11 je přehled těchto povinných polí ve formátu SAM. [21]

Tabulka 11: SAM – povinná pole. [21]

Sloupec	Pole	Typ	Regex/Rozsah	Stručný popis
1	QNAME	String	[!-?A-~]{1,254}	Jméno předlohy dotazu
2	FLAG	Int	$[0,2^{16}-1]$	Bitový FLAG
3	RNAME	String	*|[!-()+-<>-~][!-~]*	Jméno referenční sekvence
4	POS	Int	$[0,2^{31}-1]$	První pozice mapování zleva
5	MAPQ	Int	$[0,2^{8}-1]$	Kvalita mapování
6	CIGAR	String	*|([0-9]+[MIDNSHPX=])+	Cigar řetězec
7	RNEXT	String	*|=|[!-()+-<>-~][!-~]*	Odkaz na jméno páru / další read
8	PNEXT	Int	$[0,2^{31}-1]$	Pozice páru / další read
9	TLEN	Int	$[-2^{31}+1,2^{31}-1]$	Pozorována délka předlohy
10	SEQ	String	*|[A-Za-z=.]+	Sekce sekvence
11	QUAL	String	[!-~]+	ASCII Phred kvalita báze +33

Sloupec 1: QNAME (z anglického názvu **Q**uery template **NAME**). Ready/segmenty s identickými QNAME jsou považovány za to, že přicházejí ze stejné předlohy. QNAME „*“ indikuje, že informace není dostupná. V SAM souboru může read zabírat více řádků alignmentu, když je jeho alignment chimérický nebo když jde uvedeno vícenásobné mapování.

Sloupec 2: FLAG, jedná se o kombinaci bitových FLAGů. Každý bit je vysvětlen v následující tabulce 12.

Tabulka 12: SAM – FLAG, popis bitů. [21]

Bit		Popis
1	0x1	Předloha s více segmenty v sekvenování
2	0x2	Každý segment je správně zarovnán
4	0x4	nezarovnaný segment
8	0x8	Další segment v předloze je nezarovnaný
16	0x10	SEQ je reverzně komplementární
32	0x20	SEQ dalšího segmentu je reverzně komplementární
64	0x40	První segment v předloze
128	0x80	Poslední segment v předloze
256	0x100	Sekundární alignment
512	0x200	Neodpovídající filtry, jako kontrola kvality platformy/dodavatele
1024	0x400	PCR nebo optický duplikát.
2048	0x800	Doplňkový alignment

- Pro každý read v SAM souboru je nutné, aby jeden a pouze jediný řádek byl přidružen k readu splňujícím “FLAG & 0x900 == 0”. Tento řádek se nazývá primárním řádkem readu.
- Bit 0x100 označuje alignment, který nemá být použit v určitých analýzách, kdy používané nástroje jsou si vědomy tohoto bitu. Obvykle se tento bit používá pro mapování alternativních mapování, když je v SAM souboru prezentováno vícenásobné mapování.
- Bit 0x800 označuje to, že odpovídající řádek alignmentu je součástí chimérického alignmentu. Řádek s flagem 0x800 je nazývám doplňkovým řádkem.
- Bit 0x4 je jediným spolehlivým označením, kde můžeme konstatovat, že read je nezarovnán. Pokud je nastaven bit 0x4, nelze předpokládat, že lze určit hodnoty pro RNAME, POS, CIGAR, MAPQ, a bitech 0x2, 0x100 a 0x800.
- Bit 0x10 označuje to, že SEQ byla reverzně doplněna a QUAL obrácena. Pokud je bit 0x4 nenastaven, odpovídá to vláknu, ke kterému byl segment mapován. Když je bit 0x4 nastaven, znamená to, že nezarovnaný read je uložen ve své původní orientaci, v jaké přišel ze sekvenčního přístroje.

- Bity 0x40 a 0x80 reflektují pořadí readu v každé předloze, která je obsažena v použité technologii sekvenování. Pokud jsou nastaveny oba bity 0x40 a 0x80, znamená to, že je read součástí předlohy, ale není ani první ani poslední read. Pokud jsou bity 0x40 a 0x80 nenastaveny, index readu v předloze je neznámý. Toto se může stát u nelineární předlohy nebo při ztrátě této informace při zpracování dat.
- Pokud je bit 0x1 nenastaven, nelze mít žádné předpoklady o bitech 0x2, 0x8, 0x20, 0x40 a 0x80.
- Bity, které nejsou uvedeny v tabulce, jsou vyhrazeny pro budoucí použití. Při zápisu by tyto bity neměly být nastaveny a při čtení aktuálním softwarem by měly být ignorovány.

Sloupec 3: RNAME neboli jméno referenční sekvence alignmentu. Pokud jsou v záhlaví řádky s tagem @SQ, tak RNAME (pokud není '*') musí být přítomen v jednom z tagů SQ nebo SN. Nezarovnaný segment bez souřadnic má '*' v tomto poli. Nezarovnaný segment však může mít také obyčejnou souřadnici, díky které může být umístěn do požadované pozice po třídění. Pokud je RNAME '*', tak nelze určit POS ani CIGAR.

Sloupec 4: POS je první mapovaná pozice nejvíce vlevo první CIGAR operace (M, D, N, -, X). První báze v referenční sekvenci má souřadnici 1. Pro nezarovnaný read bez souřadnic je POS nastavena na 0. Pokud je POS 0, tak nelze určit RNAME ani CIGAR.

Sloupec 5: MAPQ, kvalita mapování. Popsáno v sekci 2.6. V případě, že je MAPQ nastavena na hodnotu 255, tak kvalita mapování není k dispozici.

Sloupec 6: CIGAR, řetězec CIGAR je popsán v sekci 2.7.

Sloupec 7: RNEXT, reference názvu sekvence primárního alignmentu dalšího readu v předloze. Pro poslední read je dalším readem první read v předloze. Pokud jsou v záhlaví přítomny řádky tagu @SQ, RNEXT (pokud není '*' nebo '=') musí být přítomen v jednom z tagů SQ nebo SN. Toto pole je nastaveno na '*', když není informace k dispozici nebo nastavena na '=', pokud je RNEXT identický s RNAME. Pokud toto pole nenabývá hodnoty '=' a další read v předloze má jedno primární mapování (vit bit 0x100 ve FLAGu), tohle pole je identické s RNAME na primárním řádku dalšího readu. Pokud je read '*', tak nelze určit PNEXT ani použít bit 0x20.

Sloupec 8: PNEXT, pozice primárního alignmentu dalšího readu v předloze. Pokud nejsou informace k dispozici, je toto pole nastaveno na 0. Toto pole se rovná POS primárního řádku dalšího readu. Pokud je PNEXT nastaven na hodnotu 0, tak nelze určit RNEXT ani použít bit 0x20.

Sloupec 9: TLEN, celková délka předlohy. Pokud jsou všechny segmenty mapovány na stejný odkaz, tak délka pozorované předlohy se rovná počtu bází od první levé mapované báze k poslední pravé bázi. Segment nejvíc na levé straně má označení znaménkem plus a segment nejvíce vpravo je označen znaménkem mínus. Segment uprostřed nemá definované znaménko. Hodnota tohoto pole je nastavena na 0 v případě předlohy s jedním segmentem, nebo když informace není k dispozici.

Sloupec 10: SEQ, segment sekvence. Když není sekvence uložena, tak toto pole může nabývat hodnoty '*'. Pokud toto pole nenabývá hodnoty '*', tak se délka sekvence musí rovnat součtu délek M / I / S / = / X v CIGAR řetězci. Hodnota '=' označuje, že báze je identická s referenční bází.

Sloupec 11: QUAL, ASCII kvalita báze plus 33 (stejně jako řetězec kvality v Sanger FASTQ formátu) detailněji popsáno v sekci 2.4 a 2.5. Pokud není řetězec kvality uložen, tak toto pole může nabývat hodnoty '*'. Pokud toto pole není '*', tak SEQ nesmí být '*' a délka řetězce kvality musí být rovna délce řetězce sekvence SEQ.

2.8.4 Sekce alignmentu – volitelná pole

Všechna volitelná pole mají následující formát TAG:TYP:HODNOTA, kde TAG je dvoumístný řetězec, který odpovídá /[A-Za-z][A-Za-z0-9]/. Každý TAG se může zobrazit pouze jednou v řádku alignmentu. TAGy obsahující malá písmena jsou určena pro koncové uživatele. Ve volitelném poli je TYPE jednomístný znak rozlišující malá a velká písmena, který definuje formát HODNOTY popsán v tabulce 13.

Tabulka 13: SAM – volitelná pole, typy a hodnoty tagů. [21]

Typ	Regex odpovídající HODNOTĚ	Popis
A	[!-~]	Znak výpisu.
i	[-+]?[0-9]+	Celé číslo $[-2^{31}, 2^{32}]$
f	[-+]?[0-9]*\.?[0-9]+([eE][-+]?[0-9]+)?	Formát pohyblivé řádové binární tečky
Z	[!-~]*	Řetězec výpisu včetně prázdného znaku.
H	([0-9A-F][0-9A-F])*	Bytové pole ve formátu Hex.
B	[cCsSiIf](,[-+]?[0-9]*\.?[0-9]+([eE][-+]?[0-9]+)?)+	Cele číslo nebo číselné pole.

Pro celé číslo nebo číselné pole (TYP B), označuje první písmeno typ čísel v následujícím poli odděleném čárkami. Písmeno může být jedno z ‘cCsSiIf’. Typy jsou popsány v následující tabulce 14.

Tabulka 14: SAM – volitelná pole, typy tagu typu B. [21]

Písmeno	Typ formátu
c	int8 t
C	uint8 t
s	int16 t
S	uint16 t
i	int32 t
I	uint32 t
f	float t

Během importu/export může být typ prvku změněn, pokud je nový typ kompatibilní s polem. Předdefinované nepovinné tagy jsou popsány v samostatné specifikaci „Sequence Alignment/Map Optional Fields Specification“ [22]. V této specifikaci najdeme podrobnosti o existujících standardních polí tagu a konvencí týkajících se vytváření nových tagů, které mohou být v obecném zájmu. Tagy začínající na písmena „X“, „Y“ nebo „Z“ a tagy obsahující malá písmena na všech pozicích budou vyhrazeny pro lokální použití a nikdy nebudou formálně definovány v žádné budoucí verzi této specifikace. [21]

3 Implementace

V této práci jsou použity pro porovnání výstupu SAM dva algoritmy pro zarovnání alignmentu na referenční genom, a to Needleman-Wunsch a Smith-Waterman. Funkce obou algoritmů bude popsána v samostatných bodech s problematikou, jenž má vliv na výstup do SAM souboru. Oba tyto algoritmy vytvářejí CIGAR řetězec už při zpětném krokování a k jeho převodu na zkrácenou formu používají metody třídy CIGAR jenž je universální pro oba tyto algoritmy. Všechen kód je psán v jazyce C++.

3.1 Implementace Needleman-Wunsch

Tento algoritmus je vhodný pro zarovnání kratších sekvencí, kde je jejich podobnost spíše větší a jejich délka podobná. Implementace algoritmu Needleman-Wunsch má 3 kroky, a to:

1) Inicializace
2) Naplnění matice
3) Zpětné trasování

K těmto třem krokům je také zapotřebí znát schéma ohodnocení kde, když se báze rovnají, tak platí proměnná match_score, v případě opaku platí proměnná mismatch_score a poslední, a to třetí proměnnou je gap_score, která označuje postih za vložení mezery. [1]

3.1.1 Inicializace

Nultá souřadnice sloupce / řádku je vždy 0. Poté naplníme zbývající souřadnice prvního řádku a sloupce dle vzorce:

$$M_{i,0} = i * gap\_score$$
$$M_{0,j} = j * gap\_score$$

Ukázka inicializace matice je na obrázku 8.

		A	C	G	T
	0	-2	-4	-6	-8
A	-2				
C	-4				
G	-6				
T	-8				

Obrázek 8: Needleman-Wunsch, příklad inicializace matice.

3.1.2 Naplnění matice

Pro naplnění matice M se vždy řídíme vzorcem:

$$\boldsymbol{M}_{i,j} = \boldsymbol{Max}((M_{i-1,j-1} + \boldsymbol{S}_{i,j}), (M_{i,j-1} + gap\_score), (M_{i-1,j} + gap\_score))$$

Kde souřadnici matice $\boldsymbol{M}_{i,j}$ bude přiřazena maximální hodnota pomocí funkce **Max ()** z jedné ze třech vypočítaných hodnot v matici. $\boldsymbol{S}_{i,j}$ značí v případě shody bází na souřadnicích i, j v daných sekvencích match_score a v případě neshody mismatch_score. Příklad naplnění matice na obrázku 9.

		A	C	G	T
	0	-2	-4	-6	-8
A	-2	2	0	-2	-4
C	-4	0	4	2	0
G	-6	-2	2	6	4
T	-8	-4	0	4	8

Obrázek 9: Needleman-Wunsch, příklad naplnění matice.

Na obrázku 10 lze vidět řešení v kódu, kde je zajištěno volání metody CalculateScore z obrázku 11 (metoda, kde je obsažen vzorec pro naplnění matice) a uložení posledních souřadnic do proměnných i_max a j_max. Do proměnné matrix_max se ukládá skóre z poslední souřadnice matice, které je zároveň maximálním globálním skóre zarovnání matice.

```
/*
Fill Matrix
*/
for (int i = 1; i < lena; i++) {
    for (int j = 1; j < lenb; j++) {
        int score = 0;
        score = CalculateScore(i, j);
        this->matrix_max = score;
        this->i_max = i;
        this->j_max = j;
        this->ScoringMatrix[i][j] = score;
    }
}
```

Obrázek 10: Needleman-Wunsch, příklad kódu pro naplnění matice.

```
int Needleman_Wunsch::CalculateScore(int i, int j)
{
    int max = 0;
    int diag_score = 0;
    int left_score = 0;
    int up_score = 0;
    //Set diag/left/up score
    if (this->aa[i - 1] == this->bb[j - 1]) {
        diag_score = this->ScoringMatrix[i - 1][j - 1] + this->match_score;
    }
    else {
        diag_score = this->ScoringMatrix[i - 1][j - 1] + this->mismatch_score;
    }
    left_score = this->ScoringMatrix[i][j - 1] + this->gap_score;
    up_score = this->ScoringMatrix[i-1][j] + this->gap_score;
    //Set maximum score from diag/left/up score
    if (diag_score > max) {
        max = diag_score;
    }
    if (left_score > max) {
        max = left_score;
    }
    if (up_score > max) {
        max = up_score;
    }
    return max;
}
```

Obrázek 11: Needleman-Wunsch, kód metody CalculateScore.

3.1.3 Zpětné trasování

Po kroku naplnění matice zbývá už jen zpětné trasování, které určí skutečné zarovnání, které vede k maximálnímu skóre. Trasování začíná z posledních souřadnic matice i_max a j_max a vrací se k nulté souřadnici. Při zpětném trasování se už skládá CIGAR řetězec, který bude na konci zpracován pomocí třídy CIGAR. Na obrázku 12 lze vidět kód zpětného trasování, kde jsou použity vzorce z plnění matice s tím, že nyní jsou použity k zpětnému výpočtu. Nyní se nebere nejvyšší hodnota skóre, ale nejdříve se bere v potaz podmínka pro diagonální krok (shoda/neshoda), v případě, že tento krok splní podmínku, tak se jde zpět po diagonální trase. V případě, že se podmínka nesplní, tak následuje další podmínka pro krok vlevo (Mazání), a pokud je podmínka splněna, tak se jde zpět po trase doleva. V případě, že se nesplní ani tato podmínka, zbývá pouze krok nahoru (Přidání) a ten se provede bez ohledu, jestli je podmínka splněna nebo ne.

```
/*
    Traceback
    diagonal: match/mismatch
    up:       gap in sequence 1
    left:     gap in sequence 2
*/
int m = this->i_max;
int n = this->j_max;
bool first = false; //Help to indicate fist position and don't allow next iteration to rewrite it.
while (m > 0 && n > 0)
{
    int score = 0;
    if ((m == 1 || n == 1) && first == false) {
        first = true;
        this->i_min = m;
        this->j_min = n;
    }
    if (this->aa[m - 1] == this->bb[n - 1]) {
        score = this->match_score;
    }
    else {
        score = this->mismatch_score;
    }
    //Move diag
    if (m > 0 && n > 0 && ScoringMatrix[m][n] == ScoringMatrix[m - 1][n - 1] + score)
    {
        if ((this->aa[m - 1]) != (this->bb[n - 1])) {
            this->mismatch++;
        }
        this->cigar_str = "M" + this->cigar_str;
        m--; n--;
    }
    //Move left
    else if (n > 0 && ScoringMatrix[m][n] == ScoringMatrix[m][n - 1] + this->gap_score)
    {
        this->cigar_str = "D" + this->cigar_str;
        n--;
    }
    //Move up
    else //if (m > 0 && ScoringMatrix[m][n] == ScoringMatrix[m - 1][n] + this->gap_score)
    {
        this->cigar_str = "I" + this->cigar_str;
        m--;
    }
}
```

Obrázek 12: Needleman-Wunsch, kód zpětného trasování.

3.2 Implementace Smith-Waterman

Tento algoritmus je vhodný pro zarovnání kratších sekvencí, ale na rozdíl od Needleman-Wunsche, je vhodný pro zarovnání sekvencí, mezi kterými jsou velké bázové rozdíly nebo velké délkové rozdíly. Implementace algoritmu Smith-Waterman má také 3 kroky, a to:

1) Inicializace
2) Naplnění matice
3) Zpětné trasování

K těmto třem krokům je také zapotřebí znát schéma ohodnocení kde, když se báze rovnají, tak platí proměnná match_score, v případě opaku platí proměnná mismatch_score a poslední, a to třetí proměnnou je gap_score, která označuje postih za vložení mezery. [2]

3.2.1 Inicializace

První řádek a sloupec budou vyplněny nulou. Příklad vyplnění na obrázku 13.

		A	C	G	T
	0	0	0	0	0
A	0				
C	0				
G	0				
T	0				

Obrázek 13: Smith-Waterman, příklad inicializace tabulky.

3.2.2 Naplnění matice

Naplněná matice se řídí stejným vzorcem jako u Needleman-Wunsche, ale také zde začíná odlišnost v ukládání souřadnic. Ukládá se pouze taková souřadnice, která dosahuje aktuálně nejvyššího skóre. Což je velice důležité pro další krok zpětného trasování.

```
/*
Fill Matrix
*/
for (int i = 1; i < lena; i++) {
    for (int j = 1; j < lenb; j++) {
        int score = 0;
        score = CalculateScore(i, j);
        if (score > this->matrix_max)
        {
            this->matrix_max = score;
            this->i_max = i;
            this->j_max = j;
        }
        this->ScoringMatrix[i][j] = score;
    }
}
```

Obrázek 14: Smith-Waterman, příklad kódu pro naplnění matice.

3.2.3 Zpětné trasování

U zpětného trasování dochází k výrazné změně oproti Needleman-Wunsch algoritmu. Nyní začíná zpětné trasování od pozice souřadnic, která nabyla nejvyššího skóre v průběhu naplnění matice. Zde zpětné trasování následuje vzorec kde, pokud je diagonální, levé nebo horní největší skóre rovno nule, tak došlo k ukončení zpětného trasování, i kdyby to bylo uprostřed matice. Pokud je diagonální skóre větší nebo rovno hornímu a zároveň větší nebo rovno levému skóre, tak se zpětným krokováním jde diagonálně. Pokud je horní skóre větší než diagonální a zároveň větší nebo rovno levému skóre, tak se zpětným krokováním jde nahoru. Pokud je levé skóre větší než diagonální a zároveň větší horní skóre, tak se zpětným krokováním jde nahoru. Kód zápisu CIGAR řetězce je podobný kódu Needleman-Wunsche, ale vzorec pro krokování má vlastní metodu zobrazenou na obrázku 15.

```
int Smith_Waterman::NextMove(int pos_i, int pos_j) {
    int diag = this->ScoringMatrix[pos_i - 1][pos_j - 1];
    int up = this->ScoringMatrix[pos_i - 1][pos_j];
    int left = this->ScoringMatrix[pos_i][pos_j - 1];
    if (diag >= up && diag >= left) { // Move diag
        if (diag != 0) { // 1 signals a DIAG move. 0 signals the end.
            return (int)1;
        }
        else {
            return 0;
        }
    }
    else if (up > diag && up >= left) {
        if (up != 0) { // 2 signals a UP move. 0 signals the end.
            return (int)2;
        }
        else {
            return 0;
        }
    }
    else if (left > diag && left > up) {
        if (left != 0) { // 3 signals a LEFT move. 0 signals the end.
            return (int)3;
        }
        else {
            return 0;
        }
    }
    else {
        cout << "Chyba programu" << endl;
        return 4;
    }
}
```

Obrázek 15: Smith-Waterman, kód zpětného trasování, metoda NextMove.

3.3 Implementace CIGAR

Pro vytvoření zkrácené verze CIGAR řetězce byla vytvořena třída CIGAR, jenž z řetězce např. DDMMMIMM udělá zkrácenou formu CIGAR 3M1I2M. Algoritmus prochází znak po znaku, a dokud není splněna podmínka, že prvním znakem je M, tak všechny znaky (v tomto případě D a I) zahodí. Pokud je znak např. M podruhé za sebou, tak je iterován. V případě, že po dvou znacích M bude dalším znakem např. D, tak dojde k iteraci znaku D a vypsání iterace znaku M a jeho znak samotný do proměnné typu string. V tomto aktuálním stavu by bylo v řetězci prozatím 2M. Konec zápisu CIGAR je ošetřen tak, že v případě dokončení procházení CIGAR řetězce, vypíší se pouze iterace znaku M, výpis znaku D, a I je na konci nežádoucí, a tak bude zahozen. Při každém zápisu do zkrácené formy CIGAR řetězce je připočtena délka zapisované části a po dokončení procházení CIGAR řetězce máme k dispozici i celkovou délku zkrácené verze CIGAR řetězce.

3.4 Implementace výpisu SAM souboru

Pro výpis SAM souboru je určeno několik metod třídy Output (česky výstup). První dvě metody jsou určeny pro výpis hlavičky. V první metodě se vypíší chromozomy genomu. V druhé metodě se vypíší informace o programu. Další tři metody se týkají pouze části alignmentu. První z těchto metod doplňuje a předpřipravuje data doplněna či vypočtena při iteraci přes všechny ready a jejich alignmenty do nového listu, kde jsou uloženy všechny alignmenty pro jednodušší filtrování s dodatečnými proměnnými, jako např. available, v případě, že tato proměnná je nastavena na true, tak alignment bude vypsán na výstupu a v případě hodnoty false, nebude alignment vypsán. Další proměnnou je alternative, která slouží pro zapsání informace, že existuje i jiný alignment readu se stejným skóre. Poslední přidanou proměnnou je top, která slouží pro určení alignmentu s největším skóre daného readu. Dalšími proměnnými, které se doplňují, jsou RNEXT, PNEXT a poslední z nich TLEN, kde je zapotřebí délky CIGAR řetězce. Druhá metoda pro část alignmentu slouží pouze jako filtr. V této metodě se nastaví proměnná available na false pro všechny alignmenty, které budou mít skóre menší než vstupní proměnná. Třetí, a to poslední metodou je metoda výpisu do souboru. V této metodě se projde přes všechny alignmenty předpřipraveného a profiltrovaného listu a vypíší se pouze ty, které budou splňovat podmínku, že parametr available bude true. V této části budou vypsány i dvě volitelné proměnné. První proměnnou je ar, která je výstupem proměnné alternative. V případě, že je alternative true, bude vypsáno ar:Y a v případě, že false, bude vypsáno ar:N. Druhou proměnnou je sc, která vypíše pro informaci skóre alignmentu a to ve formátu sc:i:skóre.

4 Experimenty

V této části jsou uvedeny experimenty, jejichž vstupem jsou předzpracována data FASTQ souboru, která se liší od ukázkového předzpracování. Počet záznamů v ukázkovém souboru je 428915 a počet záznamů na vstupu mé práce po předzpracování je 224320. V části předzpracování došlo k úbytku záznamů, ale pro porovnání budou použity záznamy, jež jsou jak ve výstupu této práce, tak v ukázkovém SAM souboru. Ukázkový SAM soubor byl vytvořen pomocí nástroje BWA.

4.1 Rozdíl mezi implementací Char a string pro Needleman-Wunsch

V tomto experimentu je porovnání vlivu na výkon algoritmu Needleman-Wunsche mezi implementací pracující s řetězci uložených v proměnné typu char a implementací pracující s řetězci uložených v proměnné typu string. Také místo porovnání pro získání nejvyššího skóre v metodě CalculateScore(), byla použita funkce max(). Čas bude v jednotce milisekund. Pro porovnání je přiložen i čas zpracování pomocí algoritmu Smith-Waterman. Pro tento experiment bylo pro vstup algoritmů použito 224 154 alignmentů. Prefix referenčního genomu je 0 a sufix referenčního genomu je 1. Výsledky porovnání jsou v tabulce 15.

Tabulka 15: Porovnání implementací algoritmů.

Algoritmus	Čas [ms]
Needleman-Wunsch (char)	44034 (100 %)
Needleman-Wunsch (string)	45988 (104 %)
Smith-Waterman	45411 (103 %)

Z experimentu lze vidět, že v implementaci algoritmu Needleman-Wunsche za použití proměnné typu string pro řetězce došlo k navýšení časové náročnosti o 4 % oproti implementaci algoritmu Needleman-Wunsche za použití proměnné typu char pro řetězce. U implementace algoritmu Smith-Waterman došlo k navýšení časové náročnosti o 3 % oproti implementaci algoritmu Needleman-Wunsche za použití proměnné typu char pro řetězce.

4.2 Vliv prefixu a sufixu u referenční sekvence na výsledek zpracování

Při zarovnávaní alignmentu pomocí algoritmu Neeedleman-Wunsch jsou vstupem dvě sekvence, a to sekvence alignmentu a referenční sekvence z referenčního genomu, která je načtena z pozice mapování, a to minimálně o délce sekvence read alignmentu. Tato referenční sekvence může být načtena s prefixem a/nebo sufixem bází. V tomto experimentu bude sledován vliv tohoto přidání prefixu a/nebo sufixu bází k referenční sekvenci. Pro názornost budou použity pouze dva párové řádky zpracování SAM souboru. Pro zobrazení budou zobrazeny pouze sloupce 4 až 9. Sloupce 1, 2, 3, 10 a 11 jsou pro toto porovnání vždy stejné a nebudou proto zobrazeny pro lepší přehlednost vlivů prefixu a sufixu při zarovnání alignmentu readu. V tabulce 16 jsou uvedeny výsledné výstupy experimentu.

Tabulka 16: Vliv prefixu/sufixu u referenční sekvence na výstup SAM souboru.

Počet bází prefixu a sufixu referenčního genomu	Výsledný záznam v SAM souboru.
Prefix = 0 Sufix = 1 (výchozí nastavení)	114716019 60 130M = 114716030 142 114716030 60 131M = 114716019 -142 (výchozí hodnota)
Prefix = 5 Sufix = 1	114716024 60 129M = 114716035 141 114716035 60 130M = 114716024 -141
Prefix = 0 Sufix = 5	114716019 60 127M1D2M1D1M = 114716030 144 114716030 60 130M2D1M = 114716019 -144
Prefix = 5 Sufix = 5	114716024 60 126M1D2M1D1M = 114716035 143 114716035 60 129M2D1M = 114716024 -143

Veškeré rozdíly vůči výchozí hodnotě jsou zvýrazněny červenou barvou. V prvním testu, kde je navýšen prefix na hodnotu 5 a sufix ponechán na výchozí hodnotě 1, má toto nastavení hlavní vliv na pozici zarovnání alignmentu a lze vidět také zkrácení CIGAR řetězce o jeden znak, jenž zkracuje celkovou délku překrytých párových alignmentů. V druhém testu, kde je prefix na výchozí hodnotě 0 a sufix navýšen na hodnotu 5, lze vidět hlavní vliv tohoto nastavení CIGAR řetězci, kde došlo k jeho zvětšení, které má vliv na celkovou délku překrytých párových alignmentů. V třetím a posledním testu se při zvýšení hodnoty prefixu a sufixu na 5 projevují nepříznivé vlivy obou předchozích testů. Na základě tohoto experimentu lze usoudit, že čím víc jsou si délky sekvence alignmentu a referenční sekvence rovny, tím přesnější je zarovnání alignmentu a CIGAR řetězec. Na

základě tohoto zjištění doporučuji pro zarovnávaní alignmentů pomocí Needleman-Wunsch algoritmu mít délku referenční sekvence minimálně rovnu délce sekvence alignmentu.

4.3 Porovnání výsledků Needleman-Wunsch se Smith-Waterman

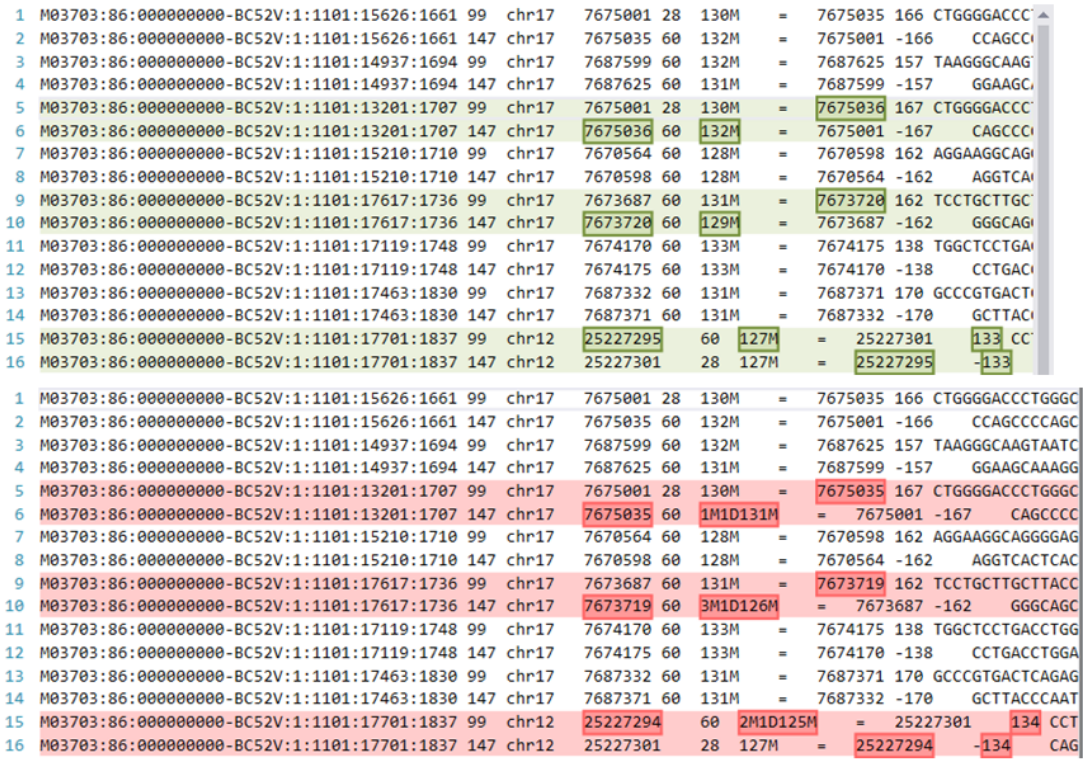
V této části experimentu je srovnání výstupu SAM souboru mezi algoritmy Needleman-Wunsch a Smith-Waterman. Pro porovnání jsou použity pouze povinné atributy SAM souboru. Výsledky celkového porovnání SAM souborů obou algoritmů jsou v tabulce 17. Na výstup do SAM souboru obou algoritmů byl aplikován filtr, kde záznam, který měl skóre alignmentu menší než 20, byl vyřazen. Pouze u SAM souboru po algoritmu Needleman-Wunsch bylo vyřazeno 166 záznamů.

Tabulka 17: Porovnání Needleman-Wunsch se Smith-Waterman.

Počet řádků záznamů Needleman-Wunch	224 154
Počet řádků záznamů Smith-Waterman	224 320
Porovnáno řádků záznamů	224 154 (100 %)
Shodné řádky záznamů	120 672 (53,83 %)
Řádky záznamů s rozdílem	103 482 (46,17 %)
Rozdíl v POS	65 860 (29,38 %)
Rozdíl v MAPQ	1 947 (0,87 %)
Rozdíl v CIGAR	67 593 (30,15 %)
Rozdíl v PNEXT	53 180 (23,72 %)
Rozdíl v TLEN	31 380 (14 %)

Při porovnání byly porovnány pouze záznamy, které měli shodné sloupce QNAME a SEQ. Při nalezení záznamu jsou porovnávány sloupce POS, MAPQ, CIGAR, PNEXT a TLEN, které jsou pro nás při tomto porovnávaní nejdůležitější. Pokud se všech pět atributů shoduje, tak je záznam pro nás shodný a pokud dochází k neshodě, tak pro větší přesnost experimentu se zaznamená, pro jaké povinné pole došlo k rozdílu dat. Jak lze vidět v tabulce 17, počet záznamů SAM souboru po algoritmu Needlman-Wunsche je 224 154 a počet záznamů SAM souboru po algoritmu Smith-Waterman je 224 320. Při porovnání obou SAM souborů bylo nalezeno všech 224 154 záznamů po algoritmu Needleman-Wunsche v SAM souboru po algoritmu Smith-Waterman. Shodných záznamů pro oba SAM soubory je 120 672 (53,83 %), což je lehce nad polovinou. Zbývajících 103 482 (46,17 %) záznamů je neshodných. V jakých sloupcích došlo k rozdílu a v jaké četnosti je zapsáno v tabulce 17. Nejčetnějším rozdílem je sloupec CIGAR, který také zapříčiní ve většině případů špatnou pozici zarovnání POS a tím, že je mnohdy i delší, takže zapříčiní také změnu hodnoty sloupce TLEN. Následně po špatném zarovnání read alignmentu dojde k tomu, že jeho párový read alignment bude mít v sloupci PNEXT zapsanou pozici POS onoho špatně zarovnaného read alignmentu. Tyto rozdíly

a jejich vliv lze vidět v příkladu vizuálního porovnání na obrázku 16, kde jsou v horní části zobrazeny výsledné záznamy Needleman-Wunsche a v spodní části jsou zobrazeny výsledné záznamy Smith-Waterman.



```
1  M03703:86:000000000-BC52V:1:1101:15626:1661 99  chr17   7675001 28  130M    =   7675035 166 CTGGGGACCC
2  M03703:86:000000000-BC52V:1:1101:15626:1661 147 chr17   7675035 60  132M    =   7675001 -166    CCAGCC
3  M03703:86:000000000-BC52V:1:1101:14937:1694 99  chr17   7687599 60  132M    =   7687625 157 TAAGGGCAAG
4  M03703:86:000000000-BC52V:1:1101:14937:1694 147 chr17   7687625 60  131M    =   7687599 -157    GGAAGC
5  M03703:86:000000000-BC52V:1:1101:13201:1707 99  chr17   7675001 28  130M    =   7675036 167 CTGGGGACCC
6  M03703:86:000000000-BC52V:1:1101:13201:1707 147 chr17   7675036 60  132M    =   7675001 -167    CAGCCC
7  M03703:86:000000000-BC52V:1:1101:15210:1710 99  chr17   7670564 60  128M    =   7670598 162 AGGAAGGCAG
8  M03703:86:000000000-BC52V:1:1101:15210:1710 147 chr17   7670598 60  128M    =   7670564 -162    AGGTCA
9  M03703:86:000000000-BC52V:1:1101:17617:1736 99  chr17   7673687 60  131M    =   7673720 162 TCCTGCTTGC
10 M03703:86:000000000-BC52V:1:1101:17617:1736 147 chr17   7673720 60  129M    =   7673687 -162    GGGCAG
11 M03703:86:000000000-BC52V:1:1101:17119:1748 99  chr17   7674170 60  133M    =   7674175 138 TGGCTCCTGA
12 M03703:86:000000000-BC52V:1:1101:17119:1748 147 chr17   7674175 60  133M    =   7674170 -138    CCTGAC
13 M03703:86:000000000-BC52V:1:1101:17463:1830 99  chr17   7687332 60  131M    =   7687371 170 GCCCGTGACT
14 M03703:86:000000000-BC52V:1:1101:17463:1830 147 chr17   7687371 60  131M    =   7687332 -170    GCTTAC
15 M03703:86:000000000-BC52V:1:1101:17701:1837 99  chr12   25227295    60  127M    =   25227301    133 CC
16 M03703:86:000000000-BC52V:1:1101:17701:1837 147 chr12   25227301    28  127M    =   25227295    -133
```

```
1  M03703:86:000000000-BC52V:1:1101:15626:1661 99  chr17   7675001 28  130M    =   7675035 166 CTGGGGACCCTGGGC
2  M03703:86:000000000-BC52V:1:1101:15626:1661 147 chr17   7675035 60  132M    =   7675001 -166    CCAGCCCCAGC
3  M03703:86:000000000-BC52V:1:1101:14937:1694 99  chr17   7687599 60  132M    =   7687625 157 TAAGGGCAAGTAATC
4  M03703:86:000000000-BC52V:1:1101:14937:1694 147 chr17   7687625 60  131M    =   7687599 -157    GGAAGCAAAGG
5  M03703:86:000000000-BC52V:1:1101:13201:1707 99  chr17   7675001 28  130M    =   7675035 167 CTGGGGACCCTGGGC
6  M03703:86:000000000-BC52V:1:1101:13201:1707 147 chr17   7675035 60  1M1D131M    =   7675001 -167    CAGCCCC
7  M03703:86:000000000-BC52V:1:1101:15210:1710 99  chr17   7670564 60  128M    =   7670598 162 AGGAAGGCAGGGGAG
8  M03703:86:000000000-BC52V:1:1101:15210:1710 147 chr17   7670598 60  128M    =   7670564 -162    AGGTCACTCAC
9  M03703:86:000000000-BC52V:1:1101:17617:1736 99  chr17   7673687 60  131M    =   7673719 162 TCCTGCTTGCTTACC
10 M03703:86:000000000-BC52V:1:1101:17617:1736 147 chr17   7673719 60  3M1D126M    =   7673687 -162    GGGCAGC
11 M03703:86:000000000-BC52V:1:1101:17119:1748 99  chr17   7674170 60  133M    =   7674175 138 TGGCTCCTGACCTGG
12 M03703:86:000000000-BC52V:1:1101:17119:1748 147 chr17   7674175 60  133M    =   7674170 -138    CCTGACCTGGA
13 M03703:86:000000000-BC52V:1:1101:17463:1830 99  chr17   7687332 60  131M    =   7687371 170 GCCCGTGACTCAGAG
14 M03703:86:000000000-BC52V:1:1101:17463:1830 147 chr17   7687371 60  131M    =   7687332 -170    GCTTACCCAAT
15 M03703:86:000000000-BC52V:1:1101:17701:1837 99  chr12   25227294    60  2M1D125M    =   25227301    134 CCT
16 M03703:86:000000000-BC52V:1:1101:17701:1837 147 chr12   25227301    28  127M    =   25227294    -134    CAG
```

Obrázek 16: Porovnání Needleman-Wunsch se Smith-Waterman.

4.4 Porovnání výsledků Needleman-Wunsch s ukázkovým SAM souborem

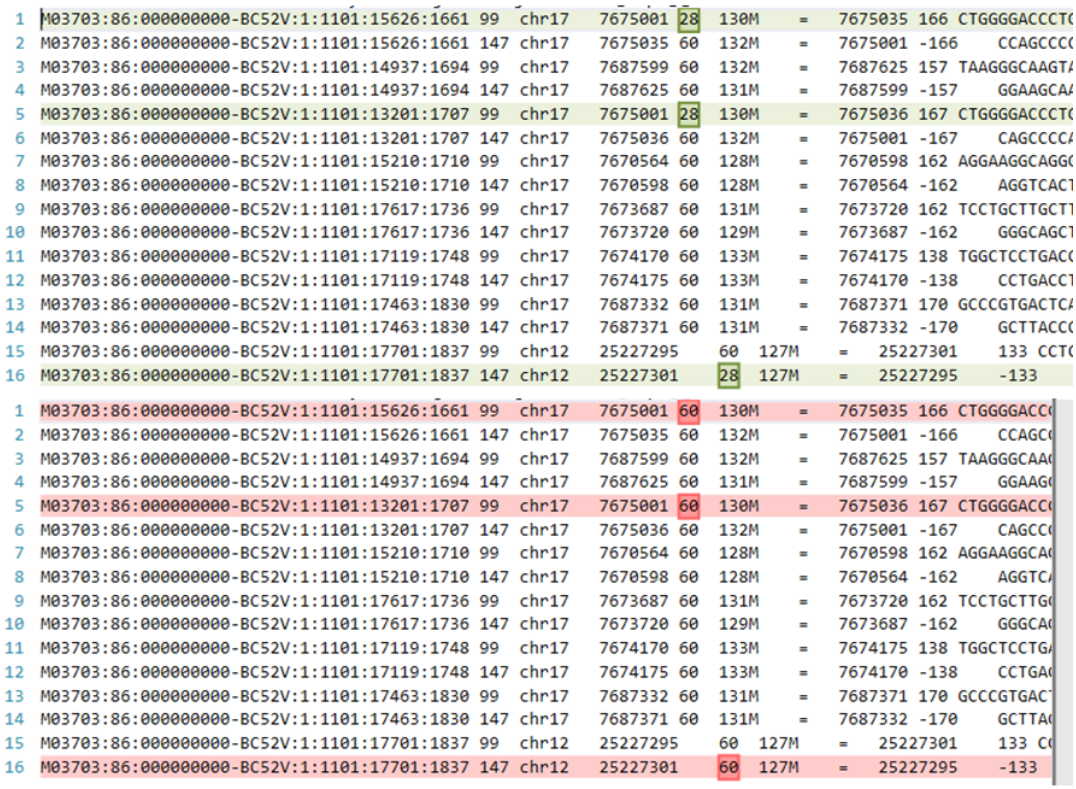
V této části experimentu je srovnání výstupu SAM souboru algoritmu Needleman-Wunsch a ukázkovým SAM souborem. Pro porovnání jsou použity pouze povinné atributy SAM souboru. Výsledky celkového porovnání SAM souborů jsou v tabulce 18. Na výstup do SAM souboru algoritmu Needlman-Wunsch byl aplikován filtr, kde záznam, který měl skóre alignmentu menší než 20, byl vyřazen, bylo tak vyřazeno 166 záznamů. U ukázkového SAM souboru byl aplikován stejný filtr záznamů.

Tabulka 18: Porovnání Needleman-Wunsch s ukázkovým SAM souborem.

Počet řádků záznamů Needleman-Wunch	224 154
Počet řádků záznamů ukázkového SAM souboru	428 915
Porovnáno řádků záznamů	224 083 (100 %)
Shodné řádky záznamů	181 016 (80,78 %)
Řádky záznamů s rozdílem	43 067 (19,22 %)
Rozdíl v POS	578 (0,26 %)
Rozdíl v MAPQ	10 467 (4,67 %)
Rozdíl v CIGAR	578 (0,26 %)
Rozdíl v PNEXT	33 821 (15,09 %)
Rozdíl v TLEN	34 062 (15,2 %)

Při porovnání byly porovnány pouze záznamy, které měli shodné sloupce QNAME a SEQ. Při nalezení záznamu jsou porovnávány sloupce POS, MAPQ, CIGAR, PNEXT a TLEN, které jsou pro nás při tomto porovnávaní nejdůležitější. Pokud se všech pět atributů shoduje, tak je záznam pro nás shodný a pokud dochází k neshodě, tak pro větší přesnost experimentu se zaznamená, pro jaké povinné pole došlo k rozdílu dat. Jak lze vidět v tabulce 18, počet záznamů SAM souboru po algoritmu Needlman-Wunsche je 224 154 a počet záznamů ukázkového SAM souboru je 428 915. Při porovnání obou SAM souborů bylo nalezeno 224 083 záznamů po algoritmu Needleman-Wunsche v ukázkovém SAM souboru. Shodných záznamů pro oba SAM soubory je 181 016 (80,78 %), což je už ucházející výsledek. Zbývajících 43 067 (19,2 %) záznamů je neshodných. V jakých sloupcích došlo k rozdílu a v jaké četnosti je zapsáno v tabulce 18. Nejčetnějším rozdílem je sloupec PNEXT a TLEN, kde pravděpodobnou příčinou v drtivé většině je chyba předzpracovaných dat, kde chybí párový read alignment. Tudíž na pozici PNEXT a TLEN je hodnota 0. Dalším sloupcem, kde je vyšší nepřesnost je MAPQ. Tato nepřesnost je způsobena zvoleným řešením výpočtu MAPQ. Zbývající sloupce CIGAR a POS mají velice nízký výskyt neshod a to 578 (0,26 %). Kdyby byla

vynechána chyba předzpracování a jiná zvolená metoda výpočtu MAPQ, tak shoda obou SAM souborů by byla v 99,74 %, což je velice uspokojivé číslo, kterého bylo dosaženo správným naimplementováním algoritmu Needleman-Wunsch a výpočtu CIGAR řetězce. Rozdíly lze vidět v příkladu vizuálního porovnání na obrázku 17, kde jsou v horní části zobrazeny výsledné záznamy Needleman-Wunsche a v spodní části jsou zobrazeny záznamy ukázkového SAM souboru.



Obrázek 17: Porovnání Needleman-Wunsch s ukázkovým SAM souborem.

5 Závěr

Teorie biologie na začátku práce nám přibližuje základní pojmy a popisuje, s jakou biologickou částí se v této práci pracovalo. Následně byly popsány formáty a standardy, jež jsou základem pro implementaci. Na vstupu programu jsou dva párové ready ze sekvenátoru a referenční genom. Tyto vstupy byly předzpracovány pro použití v mé práci. Úspěšně se povedlo implementovat algoritmy Needleman-Wunsch a Smith-Waterman pro zarovnání read alignmentů a pomocí experimentů bylo hlavně odzkoušeno chování algoritmu Needleman-Wunsche, kde se potvrdilo, že čím větší jsou rozdíly mezi délkou sekvence alignmentu a sekvence z referenčního genomu, tak tím dochází k větší nepřesnosti zarovnání, a tudíž bylo potvrzeno to, že je tento algoritmus vhodný pro zarovnání kratších sekvencí, kde je jejich délka podobná. Následně výpočet CIGAR řetězce při algoritmu Needleman-Wunsche byl téměř ve všech záznamech identický (99,74 %) s ukázkovým SAM souborem a tudíž naplnil, očekávaní vysoké přesnosti. Co se týče vypočítané mapovací kvality, ta byla také téměř ve všech záznamech identická (95,33 %) s ukázkovým SAM souborem. Je zde mnoho možností, jak vypočítat mapovací kvalitu, a proto si myslím, že tato část by byla vhodná také pro pokračování další práce, kde by došlo k porovnání různých metod výpočtu a jejich podrobnějším popsání s experimenty a dosažení tak ještě větší přesnosti mapovací kvality. Porovnání výstupů SAM souboru mezi algoritmy Needleman-Wunsche a Smith-Watermana, kde byla shoda lehce nad polovinou všech záznamů (53,83 %) ukázalo, že pro použití v mé práci je nejlepší volbou algoritmus Needleman-Wunsch, jehož výsledný SAM soubor v následném porovnání s ukázkovým SAM souborem ukázal vyšší přesnost výstupu (80,78 %). Pokračováním této práce by mohlo být také následné komprimování SAM souboru do formátu BAM, jenž by bylo finální formou souboru, který by v komprimované formě mohl ušetřit i mnoho gigabajtů.

Literatura

[1] NEEDLEMAN, Saul B. a Christian D. WUNSCH. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* [online]. 1970, **48**(3), 443-453 [cit. 2019-04-23]. DOI: 10.1016/0022-2836(70)90057-4. ISSN 00222836. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/0022283670900574

[2] SMITH, T.F. a M.S. WATERMAN. Identification of common molecular subsequences. *Journal of Molecular Biology* [online]. 1981, **147**(1), 195-197 [cit. 2019-04-23]. DOI: 10.1016/0022-2836(81)90087-5. ISSN 00222836. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/0022283681900875

[3] DNA. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-23]. Dostupné z: https://en.wikipedia.org/wiki/DNA

[4] ALBERTS, Bruce. *Molecular biology of the cell*. Sixth edition. New York, NY: Garland Science, Taylor and Francis Group, [2015]. ISBN 978-0-8153-4432-2.

[5] RUSSELL, Peter J. *IGenetics*. San Francisco: Benjamin Cummings, c2002. ISBN 0-8053-4553-1.

[6] Nucleotide. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-23]. Dostupné z: https://en.wikipedia.org/wiki/Nucleotide

[7] COGHILL, Anne M. a Lorrin R. GARSON. *The ACS style guide: effective communication of scientific information*. 3rd ed. New York: Oxford University Press, 2006. ISBN 978-0-8412-3999-9.

[8] ALBERTS, Bruce. *Molecular biology of the cell*. 4th ed. New York: Garland Science, 2002. ISBN 0-8153-3218-1.

[9] STRYER, Lubert. *Biochemistry*. 3rd ed. New York: W.H. Freeman, c1988. ISBN 9780716719205.

[10] KOČÁREK, Eduard. *Genetika: obecná genetika a cytogenetika, molekulární biologie, biotechnologie, genomika*. 2. vyd. Praha: Scientia, 2008. Biologie pro gymnázia. ISBN 978-80-86960-36-4.

[11] Chromosome. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-23]. Dostupné z: https://en.wikipedia.org/wiki/Chromosome

[12] *Vega website archive* [online]. United Kingdom: European Bioinformatics Institute, 2017 [cit. 2019-03-29]. Dostupné z: http://vega.archive.ensembl.org/Homo_sapiens

[13] FASTA format. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-23]. Dostupné z: https://en.wikipedia.org/wiki/FASTA_format

[14] *What is FASTA Format?* [online]. [cit. 2019-04-23]. Dostupné z: https://zhanglab.ccmb.med.umich.edu/FASTA/

[15] *IUPAC code table* [online]. [cit. 2019-04-23]. Dostupné z: https://web.archive.org/web/20110811073845/http://www.dna.affrc.go.jp/misc/MPsrch/InfoIUPAC.html

[16] FASTQ Format. *Maq: Mapping and Assembly with Qualities* [online]. [cit. 2019-04-23]. Dostupné z: http://maq.sourceforge.net/fastq.shtml

[17] *Qualities* [online]. [cit. 2019-04-23]. Dostupné z: http://maq.sourceforge.net/qual.shtml

[18] Phred quality score. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-23]. Dostupné z: https://en.wikipedia.org/wiki/Phred_quality_score

[19] LI, H., J. RUAN a R. DURBIN. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Research* [online]. 2008, **18**(11), 1851-1858 [cit. 2019-04-23]. DOI: 10.1101/gr.078212.108. ISSN 1088-9051. Dostupné z: http://genome.cshlp.org/cgi/doi/10.1101/gr.078212.108

[20] *SAM* [online]. [2015] [cit. 2019-04-23]. Dostupné z: https://genome.sph.umich.edu/wiki/SAM

[21] *Sequence Alignment/Map Format Specification* [online]. The SAM/BAM Format Specification Working Group, 2019 [cit. 2019-04-23]. Dostupné z: https://samtools.github.io/hts-specs/SAMv1.pdf

[22] *Sequence Alignment/Map Optional Fields Specification* [online]. The SAM/BAM Format Specification Working Group, 2019 [cit. 2019-04-23]. Dostupné z: https://github.com/samtools/hts-specs/blob/master/SAMtags.pdf

Seznam příloh

Součástí BP je DVD.

Adresářová struktura přiloženého DVD:

/bp (bakalářská práce ve formátu pdf)

/data (vstupní FASTQ soubory, ukázkový SAM soubor)

/vsb-aligner (zdrojevé kódy C++, tyto zdrojové kódy jsou také dostupné online na adrese https://github.com/mvasinek/vsb-aligner/tree/kubala)